UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 09/598,436 | 06/22/2000 | Charles Robert Moore | AT9-99-453 | 5581 |

7590 10/31/2003

BRACEWELL & PATTERSON LLP
Intellectual Property Law
P O Box 969
Austin, TX 78767-0969

| EXAMINER | |
|---|---|
| HUISMAN, DAVID J | |

| ART UNIT | PAPER NUMBER |
|---|---|
| 2183 | 6 |

DATE MAILED: 10/31/2003

Please find below and/or attached an Office communication concerning this application or proceeding.

| Office Action Summary | Application No. | Applicant(s) | |
|---|---|---|---|
| | 09/598,436 | MOORE, CHARLES ROBERT | |
| | Examiner | Art Unit | |
| | David J. Huisman | 2183 | |

*-- The MAILING DATE of this communication appears on the cover sheet with th correspondence address --*

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

1)☒ Responsive to communication(s) filed on *06 October 2003* .

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

4)☒ Claim(s) *3-18* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *3-18* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

**Application Papers**

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *22 June 2000* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

11)☐ The proposed drawing correction filed on _____ is: a)☐ approved b)☐ disapproved by the Examiner.

If approved, corrected drawings are required in reply to this Office action.

12)☐ The oath or declaration is objected to by the Examiner.

**Priority under 35 U.S.C. §§ 119 and 120**

13)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____ .

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).

a) ☐ The translation of the foreign language provisional application has been received.

15)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

**Attachment(s)**

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449) Paper No(s) _____ .
4) ☐ Interview Summary (PTO-413) Paper No(s). _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other:

## DETAILED ACTION

1. Claims 3-18 have been examined.

### *Papers Submitted*

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: #5. Amendment "A" as received on 10/6/2003.

### *Withdrawn Rejections*

3. Applicant has overcome, via amendments, the rejections set forth in the previous Office Action (paper #4), which are hereby withdrawn by the examiner.

### *Claim Rejections - 35 USC § 102*

4. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

5. Claims 3, 4, 8, 10, 11, 15, 17, and 18 are rejected under 35 U.S.C. 102(b) as being anticipated by Eickemeyer et al., U.S. Patent No. 5,377,336 (as cited in the previous Office Action and herein referred to as Eickemeyer).

6. Referring to claim 3, Eickemeyer has taught a processor comprising:

a) a plurality of registers. See column 3, lines 12-14 and note that a load instruction modifies a processor's registers. The existence of processor registers is also supported in column 4 where Eickemeyer supplies the descriptions of several instructions.

b) instruction processing circuitry that fetches an instruction sequence for execution, said instruction sequence including a load instruction and a preceding instruction that precedes said load instruction in program order. It should be realized that it is inherent that processors fetch sequences of instruction for execution, wherein every instruction (except the first and last) has a preceding and subsequent instruction. Furthermore, Eickemeyer has taught fetching instructions from cache storage and placing them in an instruction buffer. See column 6, lines 52-54.

c) said instruction processing circuitry, after fetching said instruction sequence for execution and prior to dispatching said load instruction for execution and responsive to detecting said load instruction within said fetched instruction sequence, translates said load instruction into separately executable prefetch and register operations. Recall that fetched instruction sequences are stored in the instruction buffer. The load unit (Fig.1, component 107), then scans the sequence for load instructions, and upon detecting a load instruction, a prefetch operation is performed to bring the data within the system. This is done before a load is dispatched and executed as described in column 3, lines 30-42. Therefore, a load is translated into a prefetch operation and a register operation.

d) execution circuitry that performs at least said prefetch operation out-of-order with respect to said preceding instruction to prefetch data and subsequently separately executes said register operation to place said data into a register among said plurality of registers specified by said load instruction. See the first instruction sequence in column 10 for instance, and note that the

prefetch operation (T stage in cycle 3) is separate from the register operation (P stage in cycle 10 just prior to instruction I3). And, the prefetch operation is actually executed before instruction I1, which is the instruction preceding the load instruction. This can be seen because instruction I1 is not executed until cycle 9, whereas the prefetch operation is executed in cycle 3. Therefore, the prefetch is executed before the preceding instruction (out of order). Note that after the prefetch operation, in stage "P" of the register operation, the data is written into the appropriate register. This should be realized from the definition of the "P" stage in column 5, line 50.

7. Referring to claim 4, Eickemeyer has taught a processor as described in claim 3. Eickemeyer has further taught that said execution circuitry executes said register operation in-order with respect to said preceding instruction. See the first instruction sequence in column 10. Note that the register operation (just prior to I3) still follows instruction I1, which is the preceding instruction (see column 9, lines 65-68).

8. Referring to claim 8, Eickemeyer has taught a processor as described in claim 3. Eickemeyer has further taught that said execution circuitry stores said data prefetched in response to said prefetch operation in a temporary register. See Fig.2, and note that the prefetched data is stored in entry 210 after it has been retrieved along bus 221. This entry is a register in a queue of registers (temporary storage locations).

9. Referring to claim 10, Eickemeyer has taught a method of performing a load operation in a processor having a plurality of registers, said method comprising:

a) fetching an instruction sequence for execution, said instruction sequence including a load instruction and a preceding instruction that precedes said load instruction in program order. It should be realized that it is inherent that processors fetch sequences of instruction for execution,

wherein every instruction (except the first and last) has a preceding and subsequent instruction. Furthermore, Eickemeyer has taught fetching instructions from cache storage and placing them in an instruction buffer. See column 6, lines 52-54.

b) in response to fetching said instruction sequence for execution and prior to execution of said load instruction, instruction processing circuitry detecting said load instruction within said fetched instruction sequence and translating said load instruction into separately executable prefetch and register operations. Recall that fetched instruction sequences are stored in the instruction buffer. The load unit (Fig.1, component 107), then scans the sequence for load instructions, and upon detecting a load instruction, a prefetch operation is performed to bring the data within the system. This is done before a load is dispatched and executed as described in column 3, lines 30-42. Therefore, a load is translated into a prefetch operation and a register operation.

c) performing at least said prefetch operation out-of-order with respect to said preceding instruction to prefetch data. See the first instruction sequence in column 10 for instance, and note that the prefetch operation (T stage in cycle 3) is separate from the register operation (P stage in cycle 10 just prior to instruction I3). And, the prefetch operation is actually executed before instruction I1, which is the instruction preceding the load instruction. This can be seen because instruction I1 is not executed until cycle 9, whereas the prefetch operation is executed in cycle 3. Therefore, the prefetch is executed before the preceding instruction (out of order).

d) thereafter, separately executing said register operation to place said data into a register among said plurality of registers specified by said load instruction. Note that after the prefetch

operation, in stage "P" of the register operation, the data is written into the appropriate register. This should be realized from the definition of the "P" stage in column 5, line 50.

10. Referring to claim 11, Eickemeyer has taught a method as described in claim 10. Eickemeyer has further taught executing said register operation in-order with respect to said preceding instruction. See the first instruction sequence in column 10. Note that the register operation (just prior to I3) still follows instruction I1, which is the preceding instruction (see column 9, lines 65-68).

11. Referring to claim 15, Eickemeyer has taught a method as described in claim 10. Eickemeyer has further taught that performing said prefetch operation comprises storing said data in a temporary register. See Fig.2, and note that the prefetched data is stored in entry 210 after it has been retrieved along bus 221. This entry is a register in a queue of registers (temporary storage locations).

12. Referring to claim 17, Eickemeyer has taught a processor as described in claim 3. Eickemeyer has further taught that said execution circuitry performs said prefetch operation by calculating a speculative target memory address utilizing contents of at least one register without regard for whether said contents will be modified between calculation of said speculative target memory address and performing said register operation and by thereafter initiating a fetch of said data from a memory location associated within said speculative target memory address. See column 3, lines 30-42, and note that the prefetch address is a predicted (speculative) address. This speculative address is formed through the utilization of at least one register (i.e., see Fig.2, column 7, lines 22-25, and column 8, lines 33-36, and note that the address is generated via normal address generation which involves combining values specified by the B and X registers,

and the displacement D. The values in these registers are used to generate a fetch address 218 shown in Fig.2). Furthermore, from Fig.3 it can be seen that after the predicted address is determined in block 303, the prefetch occurs at that memory address without regard of whether it will be modified or not. After the prefetch has occurred and the real address is calculated, if it is incorrect, the appropriate action will be taken (as shown in Fig.4).

13. Referring to claim 18, Eickemeyer has taught a method as described in claim 10. Eickemeyer has further taught that performing said prefetch operation comprises calculating a speculative target memory address utilizing contents of at least one register without regard for whether said contents will be modified between calculation of said speculative target memory address and performing said register operation and thereafter initiating a fetch of said data from a memory location associated within said speculative target memory address. See column 3, lines 30-42, and note that the prefetch address is a predicted (speculative) address. This speculative address is formed through the utilization of at least one register (i.e., see Fig.2, column 7, lines 22-25, and column 8, lines 33-36, and note that the address is generated via normal address generation which involves combining values specified by the D, B, and X registers. The values in these registers are used to generate a fetch address 218 shown in Fig.2). Furthermore, from Fig.3 it can be seen that after the predicted address is determined in block 303, the prefetch occurs at that memory address without regard of whether it will be modified or not. After the prefetch has occurred and the real address is calculated, if it is incorrect, the appropriate action will be taken (as shown in Fig.4).

***Claim Rejections - 35 USC § 103***

14. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

15. Claims 5 and 12 are rejected under 35 U.S.C. 103(a) as being unpatentable over Eickemeyer, as applied above.

16. Referring to claim 5, Eickemeyer has taught a processor as described in claim 3. Although Eickemeyer has disclosed executing load instructions out of order (column 12, lines 49-52), Eickemeyer has not explicitly taught that said execution circuitry executes said register operation out-of-order with respect to said preceding instruction. However, Official Notice is taken that out-of-order execution and its advantages are well known and expected in the art. With out-of-order execution, instructions can be executed as soon as their data operands are available, as opposed to in-order execution, where if a preceding instruction is stalled, then all subsequent instructions are stalled as well. Therefore, it can be seen that if the preceding instruction is stalled, out-of-order execution would allow the register operation to continue without stalling, thereby maintaining throughput. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to execute the register operation out-of-order with respect to the preceding instruction.

17. Referring to claim 12, Eickemeyer has taught a method as described in claim 10. Although Eickemeyer has disclosed executing load instructions out of order (column 12, lines 49-52), Eickemeyer has not explicitly taught that executing said register operation out-of-order

with respect to said preceding instruction. However, Official Notice is taken that out-of-order execution and its advantages are well known and expected in the art. With out-of-order execution, instructions can be executed as soon as their data operands are available, as opposed to in-order execution, where if a preceding instruction is stalled, then all subsequent instructions are stalled as well. Therefore, it can be seen that if the preceding instruction is stalled, out-of-order execution would allow the register operation to continue without stalling, thereby maintaining throughput. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to execute the register operation out-of-order with respect to the preceding instruction.

18. Claims 6, 7, 13, and 14 are rejected under 35 U.S.C. 103(a) as being unpatentable over Eickemeyer, as applied above, in view of DeLano et al., U.S. Patent No. 5,396,604 (as disclosed in the previous Office Action and herein referred to as DeLano).

19. Referring to claim 6, Eickemeyer has taught a processor as described in claim 3. Eickemeyer has not explicitly taught that said prefetch operation and said register operation have a same operation code (opcode). However, DeLano has taught the concept of a prefetch and corresponding register operation having the same opcode. See column 5, lines 20-24, and column 2, lines 48-50. It should be realized that a prefetch is implemented in the same fashion as the actual register operation (as a load instruction with a single opcode). The load is viewed by the system as a prefetch instruction only when register 0 is specified within the instruction. A person of ordinary skill in the art would have recognized that by being able to specify multiple instructions with a single opcode, the overall number of opcodes used within the system would

be reduced by 1, which would possibly result in the reduction of the amount of bits required to encode an instruction. For instance, if there were 8 instructions (not including the prefetch instruction), then the 8 instructions would be represented by 3-bit opcodes (000-111). Adding a prefetch operation with a different opcode would require 4-bit opcodes (0000-1000). By using an already existing opcode for the prefetch opcode the programmer would be able to keep the amount of opcode bits at a minimum. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Eickemeyer such that said prefetch operation and said register operation have a same operation code, as taught by DeLano.

20. Referring to claim 7, Eickemeyer in view of DeLano has taught a processor as described in claim 6. Eickemeyer has further taught that said prefetch operation and said register operation specify a same target register for said data and differ only in a value of a register operation field. See Fig.2 and note that the actual register operation comprises the opcode, D, B, X, and R fields. This operation is a subset of the overall prefetch operation, which also comprises I-ADDR, D-ADDR, P, DATA, and V fields (since data associated with prefetching is stored here). Therefore, they each specify the same target address because the target address is specified by the actual register operation (which is a subset of the prefetch operation) and they only differ in the additional fields. For instance, the additional fields which are not actually part of the register operation include a register operation field, for instance, the P field (shown in Fig.2), which is only reflects whether a prefetch occurs or not. This field is not affected by the actual register operation itself, and therefore this field is not needed by the register operation. Consequently, the register and prefetch operations differ when concerned with this field because the register operation has no value for this field while the prefetching operation is concerned with setting and

resetting this field. It should be noted that the applicant has not defined a register operation field within the claim, and therefore, any field would suffice as a register operation field.

21. Referring to claim 13, Eickemeyer has taught a method as described in claim 10. Eickemeyer has not explicitly taught translating said load instruction into prefetch and register operations having a same operation code (opcode). However, DeLano has taught the concept of a prefetch and corresponding register operation having the same opcode. See column 5, lines 20-24, and column 2, lines 48-50. It should be realized that a prefetch is implemented in the same fashion as the actual register operation (as a load instruction with a single opcode). The load is viewed by the system as a prefetch instruction only when register 0 is specified within the instruction. A person of ordinary skill in the art would have recognized that by being able to specify multiple instructions with a single opcode, the overall number of opcodes used within the system would be reduced by 1, which would possibly result in the reduction of the amount of bits required to encode an instruction. For instance, if there were 8 instructions (not including the prefetch instruction), then the 8 instructions would be represented by 3-bit opcodes (000-111). Adding a prefetch operation with a different opcode would require 4-bit opcodes (0000-1000). By using an already existing opcode for the prefetch opcode the programmer would be able to keep the amount of opcode bits at a minimum. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Eickemeyer such that said prefetch operation and said register operation have a same operation code, as taught by DeLano.

22. Referring to claim 14, Eickemeyer in view of DeLano has taught a method as described in claim 13. Eickemeyer has further taught that said prefetch operation and said register operation specify a same target register for said data and differ only in a value of a register

operation field. See Fig.2 and note that the actual register operation comprises the opcode, D, B, X, and R fields. This operation is a subset of the overall prefetch operation, which also comprises I-ADDR, D-ADDR, P, DATA, and V fields (since data associated with prefetching is stored here). Therefore, they each specify the same target address because the target address is specified by the actual register operation (which is a subset of the prefetch operation) and they only differ in the additional fields. For instance, the additional fields which are not actually part of the register operation include a register operation field, for instance, the P field (shown in Fig.2), which is only reflects whether a prefetch occurs or not. This field is not affected by the actual register operation itself, and therefore this field is not needed by the register operation. Consequently, the register and prefetch operations differ when concerned with this field because the register operation has no value for this field while the prefetching operation is concerned with setting and resetting this field. It should be noted that the applicant has not defined a register operation field within the claim, and therefore, any field would suffice as a register operation field.

23. Claims 9 and 16 are rejected under 35 U.S.C. 103(a) as being unpatentable over Eickemeyer, as applied above, in view of Konigsburg et al., U. S. Patent No. 5,931,957 (as applied in the previous Office Action and herein referred to as Konigsburg).

24. Referring to claim 9, Eickemeyer has taught a processor as described in claim 3. Eickemeyer has not explicitly taught a data hazard detector that, in response to detection of a hazard for said data, signals said processor to discard said data and said register operation. However, Konigsburg has taught that speculative instructions along a mispredicted branch path

must be flushed and their associated results discarded since any changes made to data by these instructions would cause a data hazard in that the data modified by these instructions should not be accessed by subsequent instructions. See column 6, line 66, to column 7, line 7. This type of flushing and discarding is well known and expected in the art and a person of ordinary skill in the art would have recognized that if a prefetch instruction and the associated load instruction are along a mispredicted branch path, then the data loaded by the instructions is undesired, and therefore, should be discarded. Also, the instructions should be cancelled so that they cannot complete and make any undesired changes to the system. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement, within Eickemeyer's system, a data hazard detector that, in response to detection of a hazard for said data, signals said processor to discard said data and said register operation.

25. Referring to claim 16, Eickemeyer has taught a processor as described in claim 10. Eickemeyer has not explicitly taught detecting a data hazard for said data and in response to detection of said hazard for said data, discarding said data and said register operation. However, Konigsburg has taught that speculative instructions along a mispredicted branch path must be flushed and their associated results discarded since any changes made to data by these instructions would cause a data hazard in that the data modified by these instructions should not be accessed by subsequent instructions. See column 6, line 66, to column 7, line 7. This type of flushing and discarding is well known and expected in the art and a person of ordinary skill in the art would have recognized that if a prefetch instruction and the associated load instruction are along a mispredicted branch path, then the data loaded by the instructions is undesired, and therefore, should be discarded. Also, the instructions should be cancelled so that they cannot

complete and make any undesired changes to the system. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement, within Eickemeyer's system, detecting a data hazard for said data and in response to detection of said hazard for said data, discarding said data and said register operation.

### *Conclusion*

26. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to David J. Huisman whose telephone number is (703) 305-7811. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (703) 305-9712. The fax phone number for the organization where this application or proceeding is assigned is (703) 872-9306.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

DJH
David J. Huisman
October 22, 2003

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100